Converting a DFA to a Regular Expression$_{JP}$

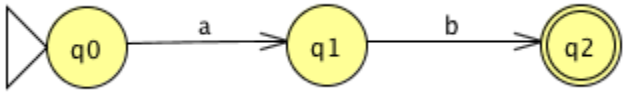
- Prerequisite knowledge:
  Regular Languages
  Deterministic Finite Automata
  Nondeterministic Finite Automata
  Regular Expressions
  Conversion of Regular Expression to Deterministic Finite Automata
  Set Theory
  JFLAP Tutorial

In this unit, we will look at the process of converting a DFA into an equivalent RE.

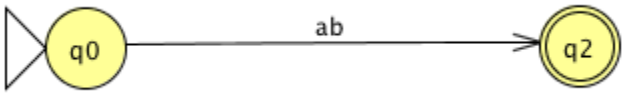
Description of DFA□RE Conversion Process

One approach to converting a DFA into an equivalent RE is to successively replace states and transitions in the DFA graph with transitions labeled with the equivalent regular expressions. Note that initially, every transition in the DFA is de facto labeled with a regular expression. If there are multiple final states, a new final state should be created with an ε-transition from each of the previous final states.

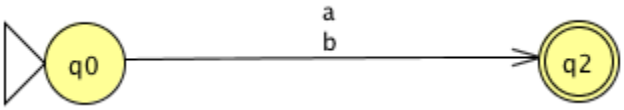
For example, consider the following FA with three states.



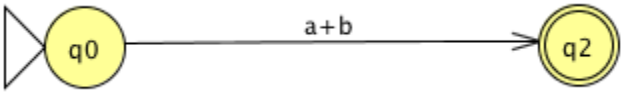
The state q1 can be replaced by a transition labeled with the regular expression **ab** representing the concatenation of a and b.



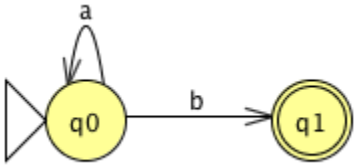
Next, consider the following FA with two states and two parallel transitions.



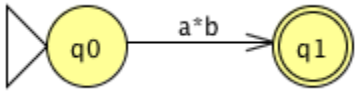
The two transitions can be replaced with a single transition labeled with the regular expression **a+b** representing the union of a and b.



Now, consider the following FA containing a transition to and from the same state.



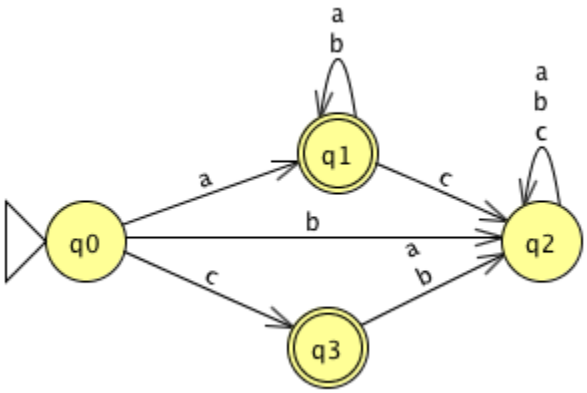
That transition indicates the Kleene star over a, which can be represented by regular expression **a***. Thus the two transitions can be replaced with a single transition labeled with the regular expression **a*b** indicating the concatenation of **a*** with **b**.



Once the FA graph has been reduced to two states (an initial state and a final state) and a single transition, the label on that transition is the regular expression equivalent to the original DFA.

Example

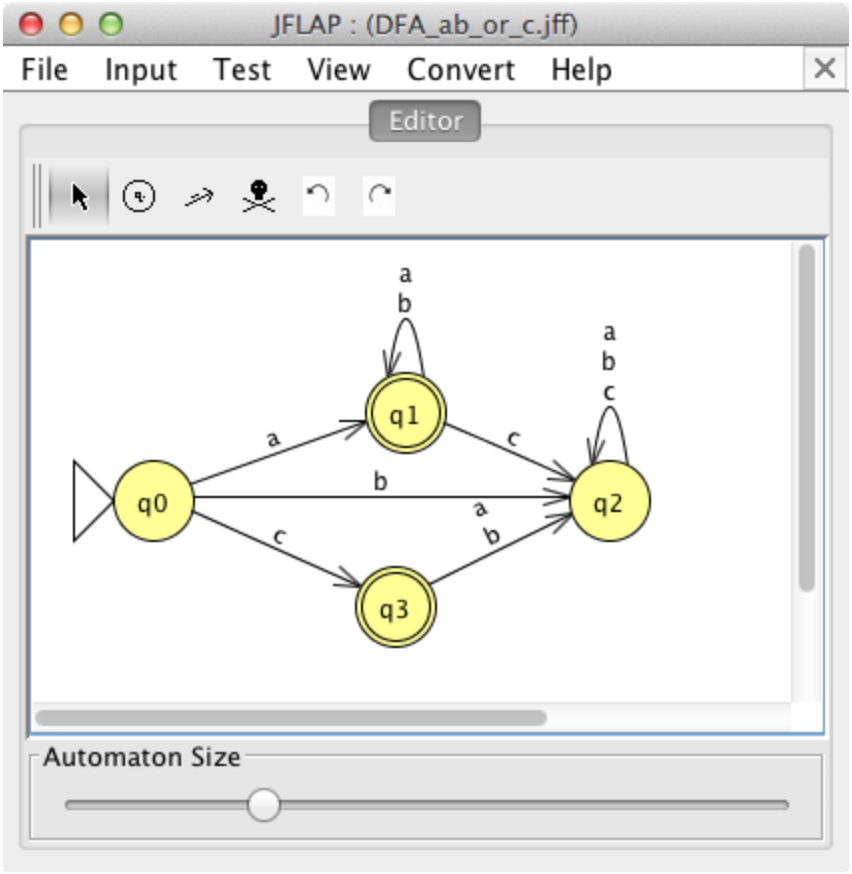
Consider the following DFA (see: `DFA_ab_or_c.jff`).



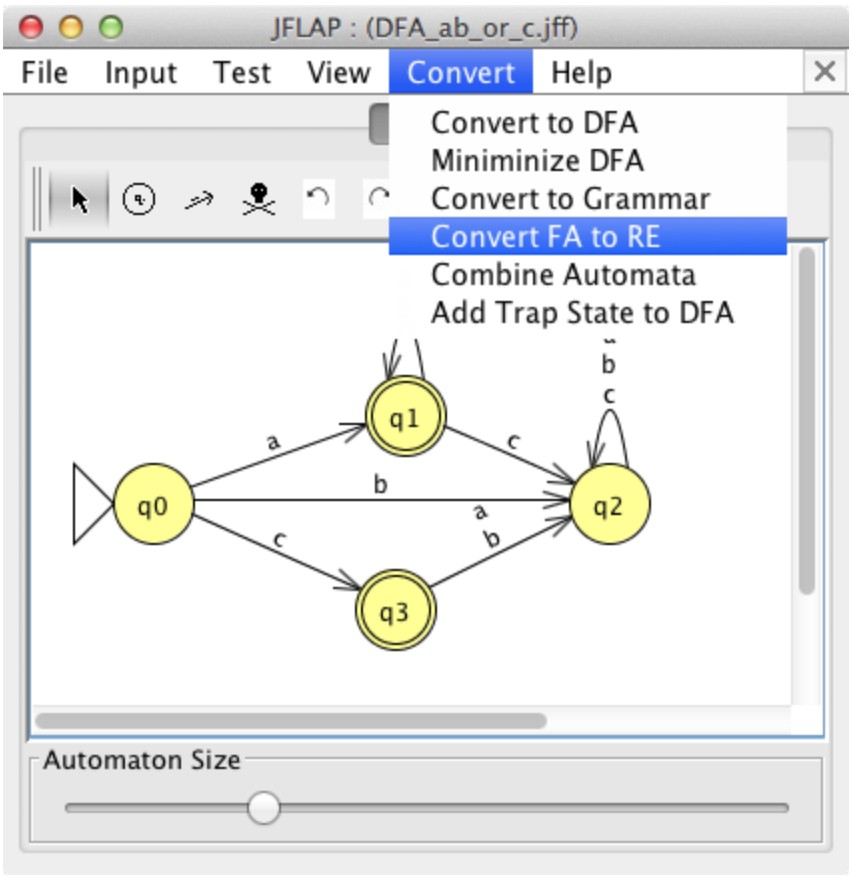
Note that because there are multiple final states, this needs to be converted to an NFA by creating a single final state with ε-transition connections from each of the previous final states. Then we can proceed with state removal and transition combination.

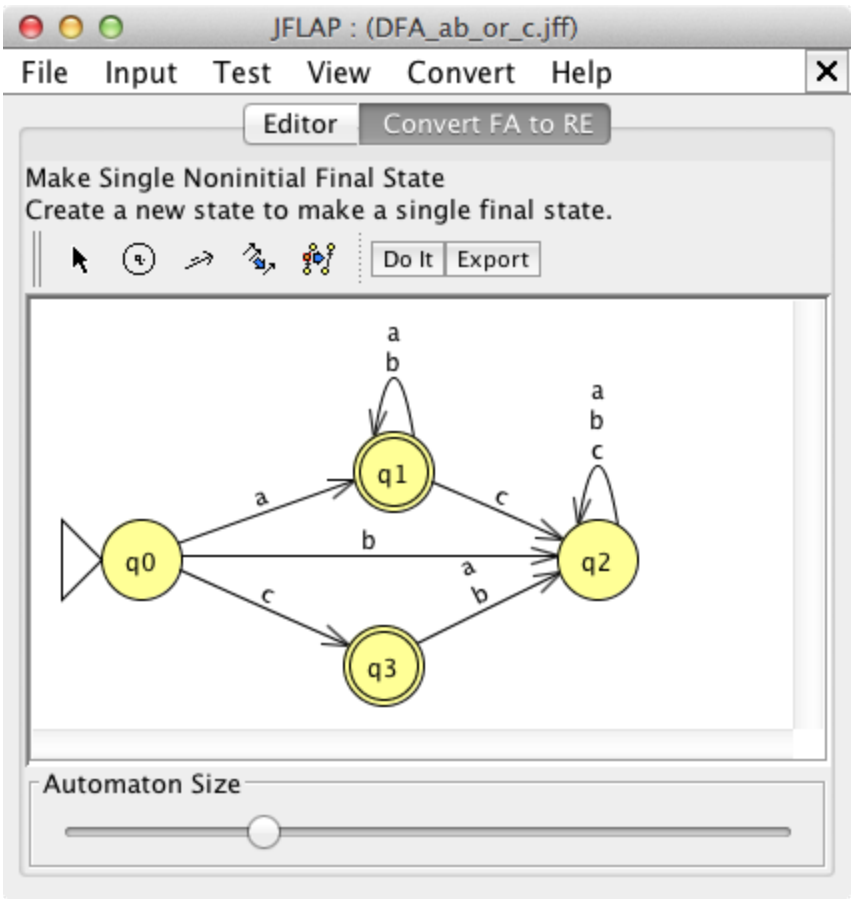
The following sequence of images depicts the path through the conversion process as provided by JFLAP.

1. First, create the DFA in JFLAP.

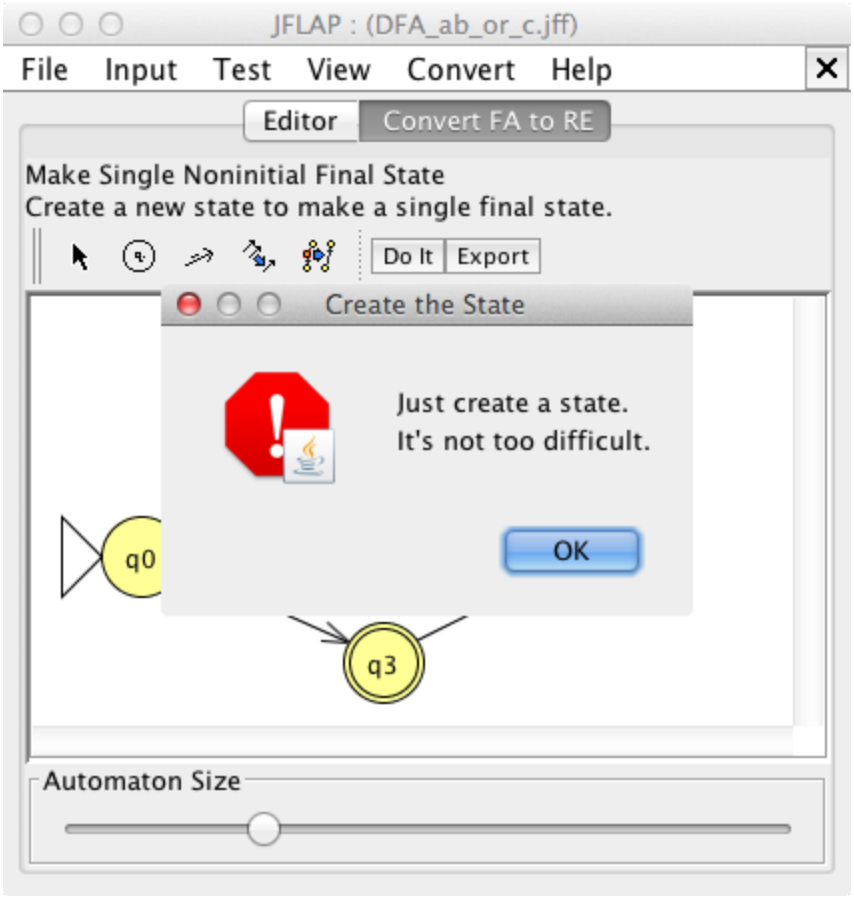


2. The conversion process begins by choosing *Convert* > *Convert FA to RE*.

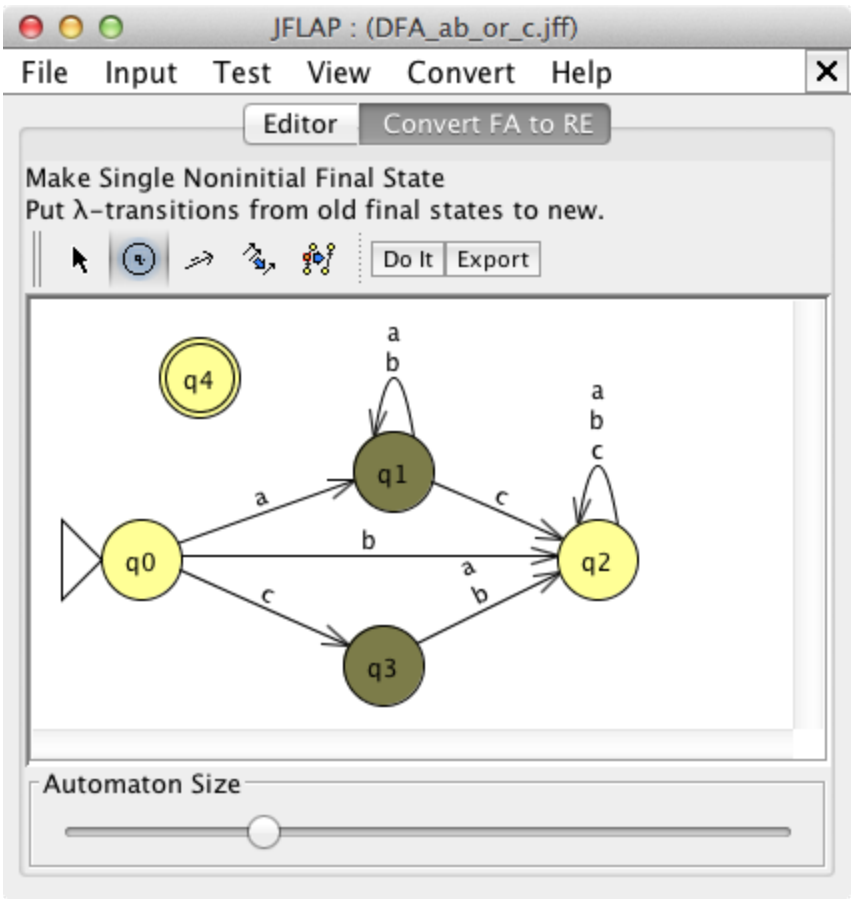

JFLAP : (DFA_ab_or_c.jff)
File
Input
Test
View
Convert
Help
Convert to DFA
Miniminize DFA
Convert to Grammar
Convert FA to RE
Combine Automata
Add Trap State to DFA
q0
q1
q2
q3
a
b
c
Automaton Size



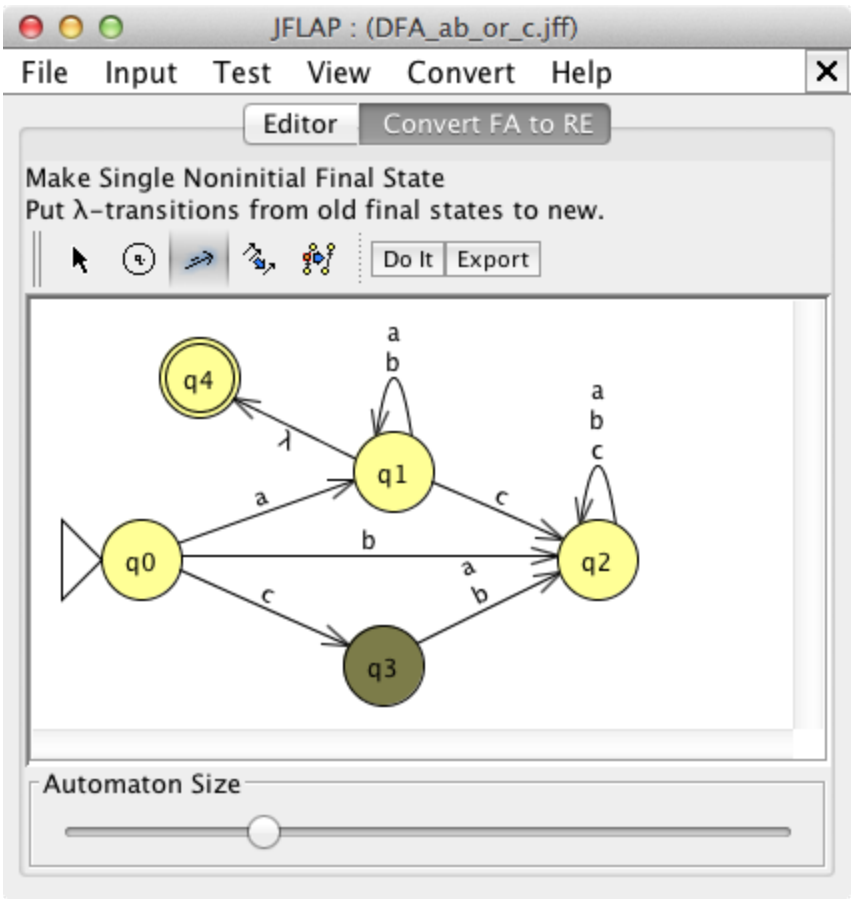
Although in general when the “Do It” button is selected JFLAP performs the indicated operation, here JFLAP instead provides a message that indicates its assumption that you already know how to accomplish the specified task.

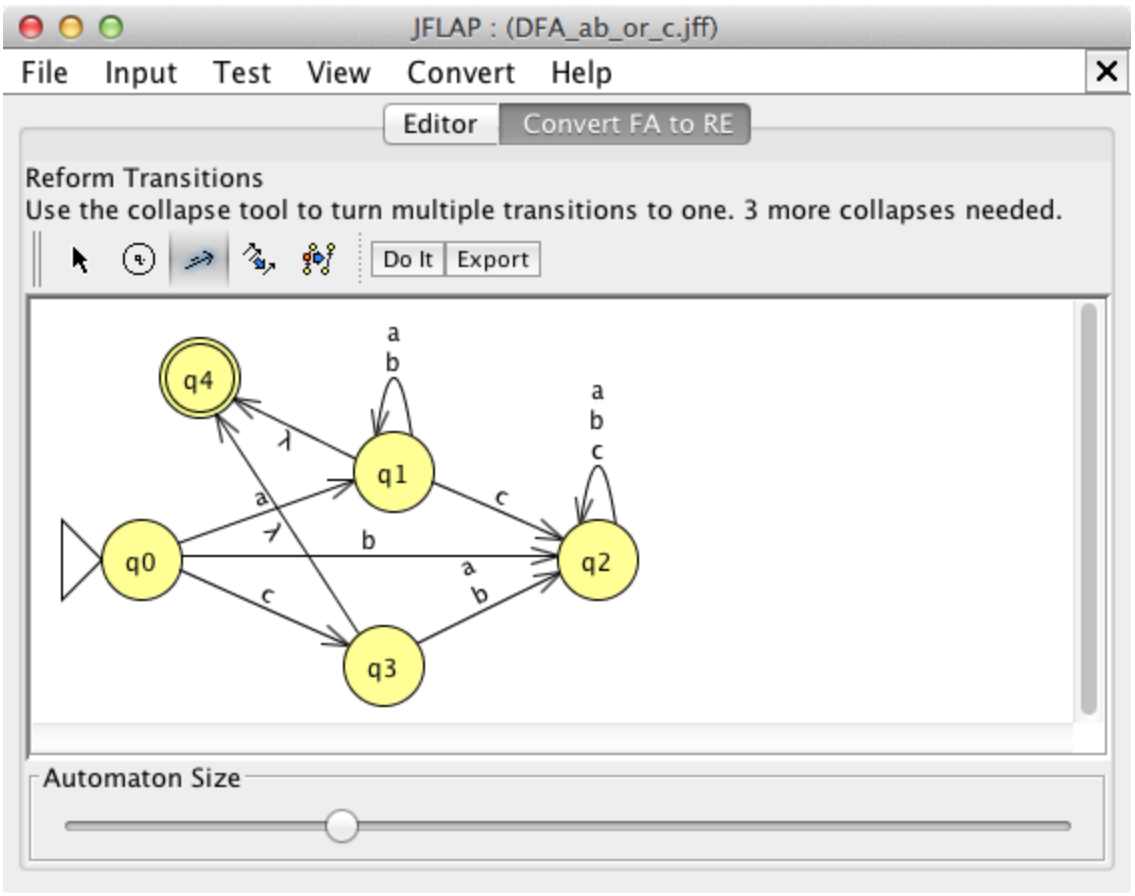


3. Manually create the new state.

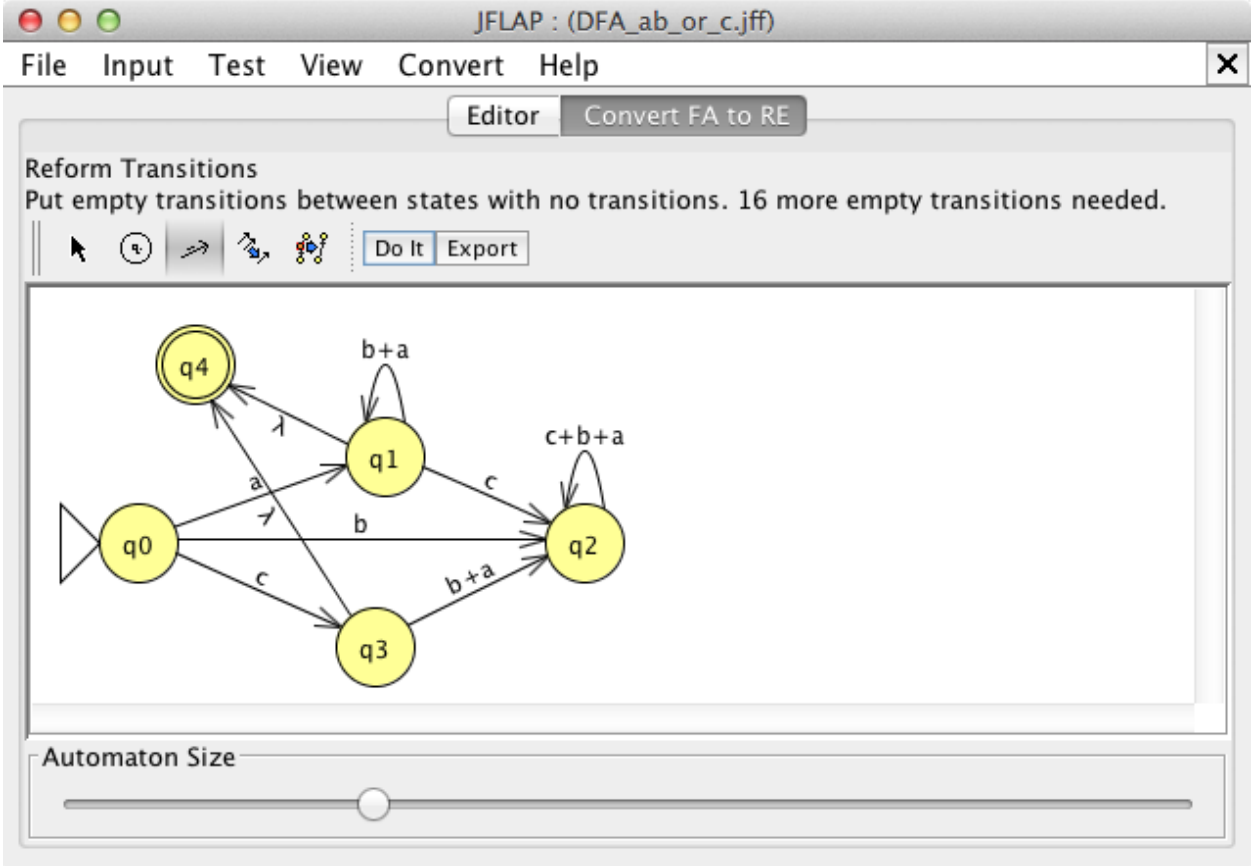


4. Add each required λ-transition.


JFLAP : (DFA_ab_or_c.jff)
File
Input
Test
View
Convert
Help
Editor
Convert FA to RE
Make Single Noninitial Final State
Put λ-transitions from old final states to new.
Do It
Export
q4
q1
q0
q2
q3
a
b
c
λ
Automaton Size

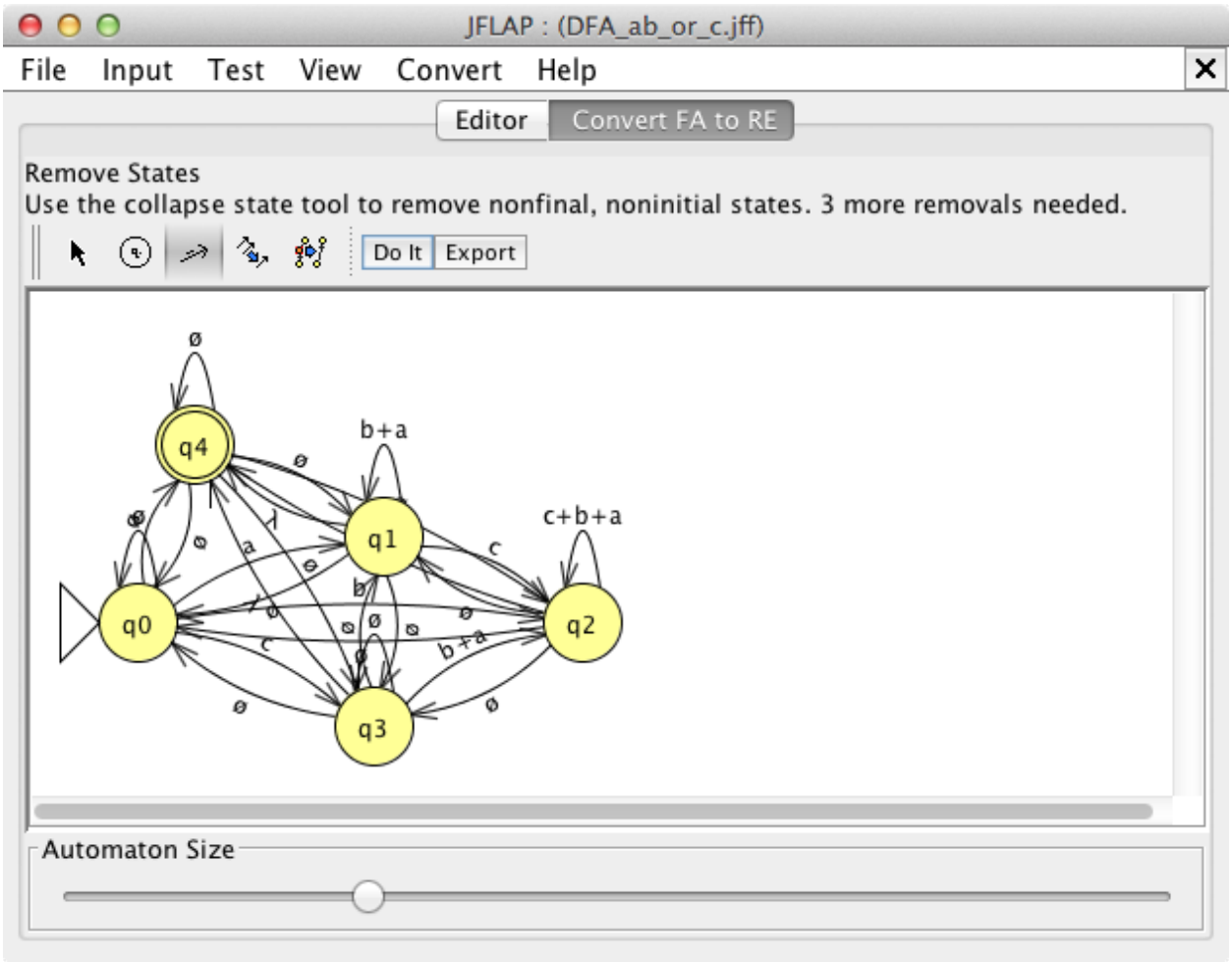


5. The next step is to collapse parallel transitions, labeling the resulting transitions with the equivalent regular expressions. Here is the result.

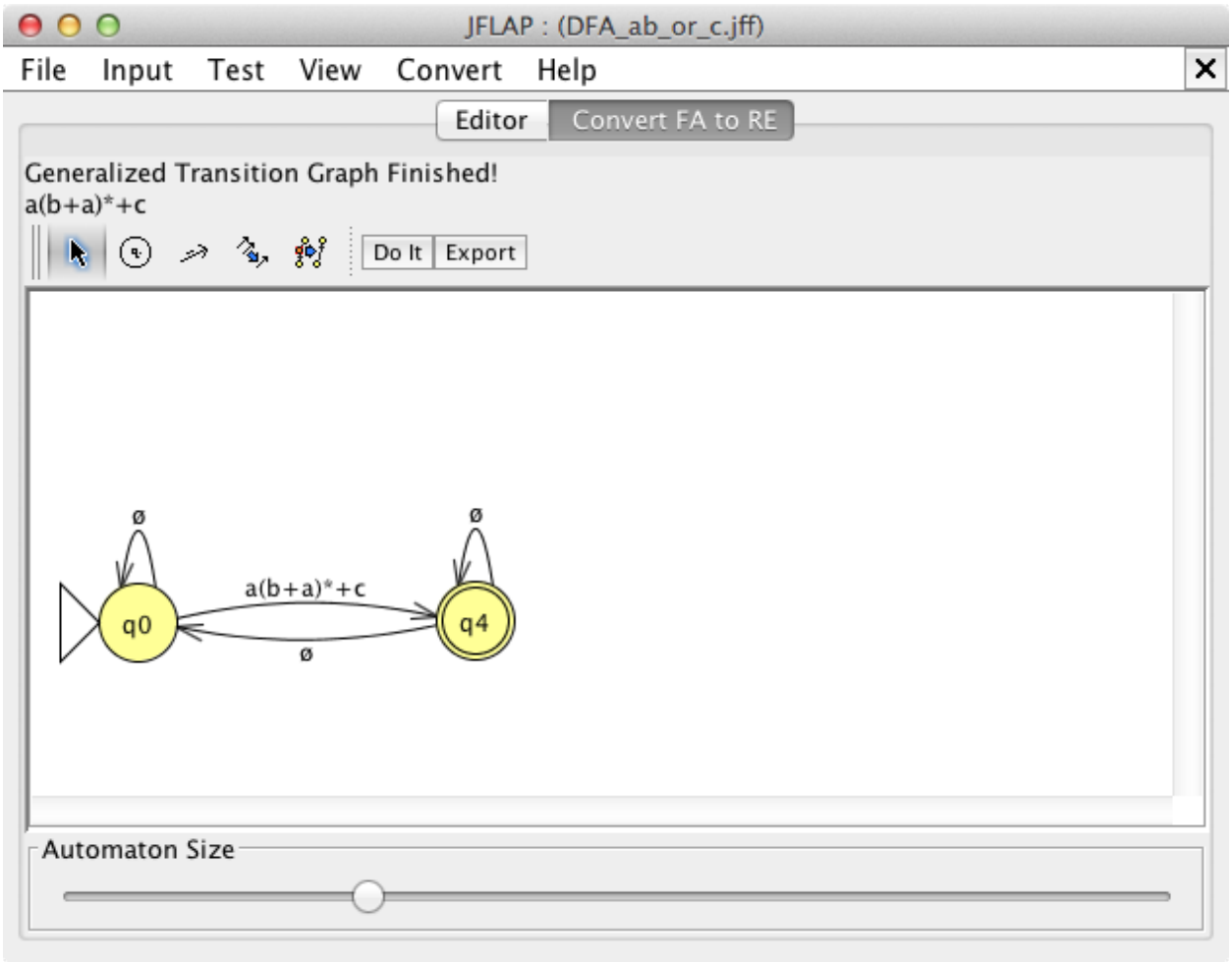


6. The next activity is adding empty transitions between every pair of states that lack an explicit existing transition, including states paired with themselves. As stated in the JFLAP tutorial, this is a necessary step because "before the conversion algorithm can work, every state must have a transition to every state in the graph, including itself."[1] Here is the result of having let JFLAP simply "Do It".

[1] http://www.jflap.org/tutorial/fa/fa2re/index.html



7. Finally, the collapse state tool provides a convenient means to eliminate states by replacing them with transitions labeled with the equivalent regular expression. The end result is shown here.



The resultant regular expression is **a(a+b)*+c** which is a specification of the language recognized by the original DFA.

Questions to Think About

1. How can you convince yourself that this regular expression is equivalent to the original DFA?
   Answer: The steps of conversion actually constitute a proof.

2. What is another regular expression that is equivalent to **a(a+b)*+c**?
   Answer: **c+a(a+b)***

3. How might that RE be derived from the same DFA starting point?
   Answer: Because union is commutative, when collapsing multiple transitions the order of operands is not significant. Thus, the two operands **c** and **a(a+b)*** could be combined either as **c+a(a+b)*** or **a(a+b)*+c**.

References

Wikipedia, *Regular Expression*
http://en.wikipedia.org/wiki/Regular_expression
[13 June 2014; Accessed 17 June 2014]

JFLAP Tutorial, *FA to Regular Expression*
http://www.jflap.org/tutorial/fa/fa2re/index.html
[Accessed 1 July 2014]